REMARKS

Claims 1-25 are pending in this application. The specification and claims 4 and 20 are being amended, as shown above. Changes made to claims 4 and 20 and the specification are shown on separate pages provided at the end of this response. Claims 4 and 20 and the specification are being amended to correct typographical errors. Applicant respectfully requests entry of the amendments and reconsideration of claims 1-25.

Status of the Drawings

The Office Action Summary notes that the drawings are objected to by the Examiner, however the Detailed Action does not indicate any specific objections. Applicant thanks the Examiner for indicating in a telephone conversation on January 16, 2003, that the objection noted in the Summary is in error and that the drawings are acceptable.

The Rejections

Claims 1-15, 18-22, and 25 have been rejected under 35 U.S.C. §102(e) as being anticipated by Gupta et al. (U.S. Patent No. 5,825,651). Claims 16, 17, 23, and 24 have been rejected under 35 U.S.C. §103(a) as unpatentable over Gupta et al. in view of Lynch et al. (U.S. Patent No. 5,515,524).

Independent Claim 1 and Dependent Claims 2-4

Claims 1-4 have been rejected under 35 U.S.C. §102(e) as being anticipated by Gupta et al. Independent claim 1 recites a method for performing a product configuration. The product configuration is associated with a configuration problem defining one or more variables and domain members associated with each variable. The method comprises receiving user input specifying at least one selected domain member, and producing a result that identifies incompatibilities between the domain members caused by the at least one selected domain member. Further, the method comprises modifying the result by detecting and eliminating incompatibilities caused solely by bounceback behavior.

Among other possible differences, the method taught by Gupta et al. does not detect and eliminate incompatibilities in a result when those incompatibilities are caused solely by bounceback behavior. Following a description of bounceback behavior, Applicant will first show that Gupta et al. does not detect incompatibilities in a result caused solely by bounceback behavior, and then will show that Gupta et al. cannot then eliminate from the result those incompatibilities caused solely by bounceback behavior.

Bounceback behavior is described in the specification by way of example (paragraphs 0005 and 0006) involving the configuration of a computer system. In the example, a set of constraints exists between processors and disk drives such that a 500 MHz processor is only compatible with 10 and 20 Gbyte disk drives, and a 700 MHz processor is only compatible with a 30 Gbyte disk drive. When the 500 MHz processor is selected in the example, constraint propagation causes the 30 Gbyte to be eliminated, and the elimination of the 30 Gbyte disk drive further causes the 700 MHz processor to be eliminated. Thus, bounceback behavior occurs where constraint propagation causes non-selected domain members of a variable to be eliminated in creating a result because another domain member of the same variable has been selected.

In the present invention, a result can be a bounceback detection bit vector (page 11 line 1), and "[e]ach member of each variable included in a given configuration problem is associated with ... a bounceback detection bit vector" (page 10 line 22 – page 11 line 1). "[E]ach bit position in a bounceback detection bit vector corresponds to a particular variable" (page 11 lines 10-11). When a user selects a first member from a variable, if constraint propagation indicates that a second member is incompatible with the first member, the second member will have a bit set in its bounceback detection bit vector, and the bit that is set will be in a position that represents the variable of the first member. Thus, if a member is of a particular variable, and a bit for that same variable is set in the bounceback detection bit vector for that member, then the result identifies a bounceback. If no other bits are set in that bounceback detection bit vector, then the incompatibility is due solely to bounceback. Accordingly, the present invention is able to evaluate the result for each member to detect whether an incompatibility for that member is due solely to bounceback behavior.

Gupta et al. does not detect incompatibilities in a created result caused *solely* by bounceback behavior. In the method of Gupta et al., selecting a member of a first set of parts will cause a part-to-part relationship, such as a removes relationship, to be enforced on a second set of parts (col. 6 lines 21-28). The two sets of parts do not need to be of a same variable, for example, selecting a sun-roof for an automobile will cause the exclusion of a roof-top antenna (col. 6 lines 39-42). A part that is removed from a configuration is noted by a corresponding bit in a removed vector 808 (rVec) (Col. 9 lines 55-66). However, the bit in the removed vector 808 does not indicate why the corresponding part has been removed from the configuration. Nothing in Gupta et al. facilitates the identification or detection of bounceback behavior as the cause of the bit having been set in the removed vector 808. Therefore, Gupta et al. cannot identify, or detect, incompatibilities in the removed vector 808 caused *solely* by bounceback behavior.

Additionally, Gupta et al. does not eliminate incompatibilities in the result caused solely by bounceback behavior. Since Gupta et al. does not teach a method of detecting incompatibilities in the result that are caused solely by bounceback, Gupta et al. cannot eliminate incompatibilities in the result caused solely by bounceback behavior. Since Gupta et al. cannot modify a result by detecting and eliminating incompatibilities caused solely by bounceback behavior, Gupta et al. does not anticipate claim 1.

For at least these reasons provided, independent claim 1 is not anticipated by Gupta et al. Accordingly, claims 2-4 depending from claim 1 are also novel over Gupta et al. Applicant further notes that claim 2 is additionally novel in that it recites generating a configuration page based on the modified result so that domain members identified as being incompatible due to bounceback behavior are not marked as conflicted choices on the configuration page. Since Gupta et al. does not teach generating a configuration page on which incompatible domain members are *not* marked as conflicted choices, Gupta et al. cannot anticipate claim 2. Applicant requests that the Examiner withdraw the rejections of claims 1-4 under 35 U.S.C. §102(e).

Independent Claim 5 and Dependent Claims 6 and 7

Claims 5-7 have been rejected under 35 U.S.C. §102(e) as being anticipated by Gupta et al. Independent claim 5 recites a system for performing a product configuration associated with a configuration problem defining one or more variables and domain members associated with each variable. The system comprises a configuration engine and a bounceback detection module. The configuration engine is adapted to receive user input specifying at least one selected domain member and to produce a result that identifies incompatibilities between the domain members caused by the at least one selected domain member. The bounceback detection module is adapted to modify the result by detecting and eliminating incompatibilities caused solely by bounceback behavior.

As noted with respect to claim 1, since Gupta et al. does not teach modifying a result by detecting and eliminating incompatibilities caused solely by bounceback behavior, Gupta et al. cannot teach a bounceback detection module adapted to such a function. Therefore, independent claim 5 is not anticipated by Gupta et al. Thus, claims 6 and 7 depending from claim 5 are also novel over Gupta et al. Applicant therefore requests that the Examiner withdraw the rejections of claims 5-7 under 35 U.S.C. §102(e).

Independent Claim 8

Claim 8 has been rejected under 35 U.S.C. §102(e) as being anticipated by Gupta et al. Independent claim 8 recites a method that comprises at least the limitations of claims 1 and 2 discussed above. Thus, for the reasons provided above with respect to claims 1 and 2, claim 8 is also novel over Gupta et al. Applicant requests that the Examiner withdraw the rejection of claim 8 under 35 U.S.C. §102(e).

Independent Claim 9 and Dependent Claims 10-20

Claims 9-15 and 18-20 have been rejected under 35 U.S.C. §102(e) as being anticipated by Gupta et al., and claims 16 and 17 have been rejected under 35 U.S.C. §103(a) as unpatentable over Gupta et al. in view of Lynch et al. Independent claim 9 recites a method for detecting bounceback behavior associated with a configuration problem defining one or more variables and domain members associated with each variable. The method comprises: receiving a domain member selection for a particular variable; setting a bounceback detection bit vector associated with each non-selected domain member of the particular variable so that each of those bounceback detection bit vectors indicates bounceback behavior; setting an elimination flag associated with each non-selected domain member of the particular variable so that each of those elimination flags indicates that its associated domain member is tentatively eliminated; setting the bounceback detection bit vector of the eliminated domain members to indicate which variable caused their elimination; and setting the elimination flag of each of the eliminated members.

As noted above, Gupta et al. does not address bounceback detection. Accordingly, Gupta et al. does not teach setting a bounceback detection bit vector associated with each non-selected domain member so that each of the bounceback detection bit vectors indicates bounceback behavior. Likewise, Gupta et al. does not teach setting an elimination flag associated with each non-selected domain member of the particular variable so that each of those elimination flags indicates that its associated domain member is tentatively eliminated. More particularly, Gupta et al. does not disclose that a domain member can be tentatively eliminated. Additionally, Gupta et al. does not teach setting a bounceback detection bit vector of an eliminated domain member to indicate which variable caused its elimination. Although, as noted above, a bit in a remove vector 808 can indicate that a first element has been removed from a configuration, there is nothing in Gupta et al. that correlates that bit to a second element who's selection caused the removal of the first element. Lastly, Gupta et al. does not teach setting an elimination flag of each of the eliminated members. For at least these reasons claim 9 is not anticipated by Gupta et al. Thus, claims 10-15 and 18-20 depending from claim 9 are also not anticipated by Gupta et al. Applicant therefore requests that the Examiner withdraw the rejection of claim 9-15 and 18-20 under 35 U.S.C. §102(e).

Claims 16 and 17 have been rejected under 35 U.S.C. §103(a) as being unpatentable over Gupta et al. in view of Lynch et al. Applicant asserts that Lynch et al.

does not cure the deficiencies of Gupta et al. discussed above with respect to claim 9 and therefore both claims 16 and 17 are patentable over the combination of Gupta et al. and Lynch et al.

However, even if, *arguendo*, Gupta et al. does anticipate claim 9, the combination of Gupta et al. and Lynch et al. does not teach or suggest every limitation of claims 16 and 17. With respect to claim 16, the Examiner asserts that Lynch et al. "teaches identifying a join corresponding to a disjoin; logically ANDing the bounceback detection bit vectors" and "copying the resulting bounceback detection bit vector to the bounceback detection bit vector associated with the eliminated domain member" (paper #2 page 5). The Examiner makes essentially the same assertion with respect to claim 17 (paper #2 page 5). However, Applicant was unable to find any teaching in Lynch et al. of identifying a join corresponding to a disjoin, bounceback detection bit vectors, or eliminated domain members. Applicant requests that the Examiner show the correlations between the limitations of these claims and the prior art in greater specificity in the next Office Action if these rejections are maintained, as citing to whole columns and whole sections of the references do not make such correlations clear.

Lastly, even if, *arguendo*, Gupta et al. does anticipate claim 9, an even if the combination of Gupta et al. and Lynch et al. does teach or suggest every element of claims 16 and 17, there is no motivation to combine the references. Applicant disagrees with the Examiner's position that it would have been obvious to combine the references "in order to provide a method of operating a telecommunications network wherein a plurality of service modules are configured for processing service requests concerning corresponding communications services. The service request prompts a selected service module to develop network instructions usable by the network for implementing the corresponding services. These instructions are then provided to the network elements for providing the requested services" (paper #2 page 5). Applicant contends that the provided motivation bears no relevance to either Gupta et al. or Lynch et al.

Accordingly, claims 16 and 17 are patentable over the combination of Gupta et al. and Lynch et al. and Applicant requests that the Examiner withdraw the rejections of claims 16 and 17 under 35 U.S.C. §103(a).

<u>Independent Claim 21 and Dependent Claims 22-25</u>

Claims 21, 22, and 25 have been rejected under 35 U.S.C. §102(e) as being anticipated by <u>Gupta et al.</u>, and claims 23 and 24 have been rejected under 35 U.S.C. §103(a) as being unpatentable over <u>Gupta et al.</u> in view of <u>Lynch et al.</u> Independent claim 21 recites a method for detecting and eliminating bounceback behavior associated with a configuration problem. Claim 21 recites at least each of the limitations discussed above with respect to claim 9 and is novel over <u>Gupta et al.</u> for at least the same reasons. Claims 22-25 depending from claim 21 are therefore also novel over <u>Gupta et al.</u> Additionally, claims 23 and 24 recite essentially the same limitations as claims 16 and 17 and are patentable over the combination of <u>Gupta et al.</u> and <u>Lynch et al.</u> for essentially the reasons provided with respect to claims 16 and 17. Accordingly, Applicant requests that the Examiner withdraw the rejections of claims 21, 22, and 25 under 35 U.S.C. §102(e) and the rejections of claims 23 and 24 under 35 U.S.C. §103(a).

All pending claims are allowable and Applicant respectfully requests a Notice of Allowance from the Examiner. Should the Examiner have questions, the Applicant's undersigned attorney may be reached at the number provided.

Respectfully submitted,

Mark J. Wihl et al.

Date: 3/7/03 By: [illegible]

Robert D. Hayden, Reg. No. 42,645
Carr & Ferrell *LLP*
2225 East Bayshore Road, Suite 200
Palo Alto, CA 94303
Phone (650) 812-3465
Fax (650) 812-3444

Version with Markings to show Changes Made

IN THE SPECIFICATION:

Paragraph 0049 on page 21 is being modified as follows:

Next, constraint K3 is propagated, with the result that the 128 Mbyte Memory is eliminated. This elimination occurs because both the 30 Gbyte and the 40 Gbyte DiskDrives have been eliminated. Thus, a join corresponding to a conjunction [is] can be derived from the given constraints. This join could be stated logically: if both of the DC2 DiskController and the 700 MHz CPU are selected, then the 128 Mbyte Memory cannot be selected. To propagate this join, the bounceback detection process logically ORs the bounceback detection bit vector for the 30 Gbyte (eliminated by selection of the DC2 DiskController) and 40 Gbyte (eliminated by selection of the 700 MHz CPU) DiskDrives thereby yielding a bounceback detection bit vector of #0101 for the 128 Mbyte Memory. The elimination flag associated with the 128 Mbyte Memory is also set.

IN THE CLAIMS:

Claims 4 and 20 are being amended as follows:

4. (Amended) The method of claim 1, wherein the method is implemented by a set of software instructions running on a computer.

20. (Amended) The method of claim 9, wherein the method is implemented by a set of software instructions running on a computer.